Submitted, Int'l Conf on Formal Modeling and Analysis of Timed Systems (FORMATS) 2019
http://www.cds.caltech.edu/~murray/preprints/pm19-formats_s.pdf

Contracts of Reactivity*

Tung Phan-Minh[1] and Richard M. Murray[2]

[1] Department of Mechanical Engineering
tung@caltech.edu
[2] Department of Control and Dynamical Systems
California Institute of Technology, Pasadena CA 91125, USA

Abstract. We present a theory of contracts that is centered around reacting to failures and explore it from a general assume-guarantee perspective as well as from a concrete context of automated synthesis from linear temporal logic (LTL) specifications, all of which are compliant with a contract metatheory introduced by Benveniste et al. We also show how to obtain an automated procedure for synthesizing reactive assume-guarantee contracts and implementations that capture ideas like optimality and robustness based on assume-guarantee lattices computed from antitone Galois connection fixpoints. Lastly, we provide an example of a "reactive GR(1)" contract and a simulation of its implementation.

Keywords: Design by contracts · Modeling uncertainties · Formal specifications · Reactive synthesis.

1 Introduction

Automating correctness and improving productivity motivate the development of formal contracts in the compositional design of large and complex engineering systems (e.g., [20]). Abstractly speaking, a contract [5] is a sufficiently unambiguous specification of a system that allows for a certain level of freedom in *implementation*. Considering the collaborative nature of modern design and manufacturing, a contract must also clearly describe the requirements of the *environment* that the system being specified is intended to operate in so that when the system and its environment are *interconnected*, i.e., allowed to interact, the desired behavior materializes.

A premise central to the field of formal methods is the idea that the mathematical model being verified is the "right" abstraction of the system in question. Oftentimes, especially for physical systems, this is not the case since the added details would make the model too large to be verified efficiently, if at all [13]. Model uncertainties, environmental disturbances, simplifying assumptions etc. must be accounted for separately, often using heuristics. In the reactive synthesis setting [6], attempts have been made to automatically generate systems that satisfy specifications with some measure of robustness to certain classes of

* Supported by DENSO International America, Inc., the Jack Kent Cooke Foundation, and the NSF VeHICaL project.

uncertainties. For instance, [17] exploits "locality" to compute modal μ-calculus fixpoints [3] that enable patching system strategies in the presence of updated information about the game graph. Similarly, [10] discusses a way to recover from a finite number of unexpected actuation or sensing errors with pre-computed safe strategies that attempt to bring the system back to the nominal control trajectory. In addition to making specific assumptions on the environment, these methods also rely on the fact that such recovery strategies are always feasible for the original set of objectives. On the specification side of things, [4] represents an elementary and direct means to add "reparation" handling to contract automata. Unfortunately, the obligation to define contract states and transitions explicitly does not make the automaton approach amenable to expressing and extracting complex properties.

Consider a standard *assume-guarantee* style design specification φ_S for a parking garage system S operated by robot valets of the form $\varphi_S := A \Rightarrow G$ (see [24] for a concrete specification in TLA$^+$), where A encodes a set of input constraints or *assumptions* such as garage topology, quantity of the valets, as well as their quality, such as how fast they can park and retrieve cars, and G consists of *guarantees* the system must provide in the event that the constraints in A are satisfied, such as an upper bound on the maximum wait times for customers. Under this assume-guarantee framework, any system that satisfies the requirement

> *unless a constraint in A is violated, all guarantees in G are provided*

is said to implement the contract. We argue that this formalism often results in a system that is

1. *fragile*: this contract is not robust to disturbances: if a car runs out of fuel and thereby invalidates the assumption on the performance of the valet assigned to it, are other valets suddenly allowed to indiscriminately or even completely abandon their responsibilities? According to φ_S, the answer is yes even though the "intuitively correct" answer is that they should not.
2. *underpromising*: the guarantee must often be very conservative when it must hold against the worst case assumptions as well the most forgiving ones: consider an abnormally slow valet biasing the guaranteed worst wait time.
3. *maladaptive*: since there is only one assumption and guarantee pair (A, G), it is not possible to adapt to changes in A nor to update guarantees in G. Even if more pairs are allowed, there is no clear procedure on how to specify controlling or switching between specific assumptions and guarantees when such an opportunity arises.

To address these issues, we propose a contract formalism that explicitly takes into account the notion of uncertainty in the system being modelled and emphasizes its obligation to adapt to possible changes in the behavior of the environment. The structure of the paper is as follows: in the first part of Section 2, a standard formal definition of assume-guarantee contracts is given as a point of reference. In the second part of Section 2, the idea of "generative effects" is introduced and

used to motivate our framework. In Section 3, our theory of reactive contracts is presented and contrasted with the non-reactive formalism. In Section 4, we specialize this theory to the context of GR(1) games and derive a procedure for formulating and synthesizing their reactive contracts. In Section 5, we apply results from Section 4 to a concrete example/simulation and explore the notion of optimality and robustness.

2 Contracts and Effects

2.1 Assume-guarantee contracts

For the sake of conciseness, we will only cover assume-guarantee contracts defined over a common set of Boolean variables $\mathcal{V}$ called the *alphabet*. One will see that a more detailed variant with local alphabets where a variable can also be labelled either as input or output is indeed more expressive (e.g., the notion of closedness/openness can be directly defined) but not required to introduce the notion of reactivity. For any set X, let X^ω be the set of infinite sequences generated from X, namely $\{\langle x_i \rangle_{i=0}^{\infty} \mid x_i \in X\}$, X^* be the free monoid on X, and 2^X be the powerset of X. Define X^∞ as $X^* \cup X^\omega$. In accordance with the metatheory presented in [5], which will, from this point on, be referred to as "the metatheory", we term any pair of collections of *environments* and *implementations* a contract. The theory of assume-guarantee contracts is a model of this metatheory and can be described as follows.

Definition 1 (Behaviors and Assertions). *A behavior σ is an element of $\mathcal{B} \coloneqq (2^\mathcal{V})^\omega$. An assertion A is a subset of $\mathcal{B}$, namely, $A \in 2^\mathcal{B}$.*

We lift the set of all assertions $2^\mathcal{B}$ to a Boolean algebra by defining a unary operator $\neg$ and two binary operators $\wedge, \vee$ on it in the standard way: if A, A_1, A_2 are assertions, then $\neg A \coloneqq \mathcal{B} \setminus A$, $A_1 \vee A_2 \coloneqq A_1 \cup A_2$, $A_1 \wedge A_2 \coloneqq A_1 \cap A_2$. The induced partial ordering relation $\leq$ on $2^\mathcal{B}$ is simply the subset relation $\subseteq$. Additionally, we define the secondary binary operator $\Rightarrow$ in $A_1 \Rightarrow A_2$ as a shorthand for $\neg A_1 \vee A_2$.

A *component* M is an assertion designated as such. Usually, the assertion characterizing a component is restricted to a subset of $\mathcal{V}$, which can be thought of as input/output variables associated with that component. This is due to the fact that most components are intended to be interconnected with others. If M_1 and M_2 are components, the *interconnection* binary operator $\oplus$ is defined by $M_1 \oplus M_2 \coloneqq M_1 \wedge M_2$. It is clear that $\oplus$ is commutative and associative, and that the set of all components is closed under it. We note that depending on how the variables and assertions making up the components being interconnected are defined, $\oplus$ can assume the meaning of either a parallel, series, coproduct, or feedback connection [8]. In fact, the definition of contracts (and the corresponding algebra) restricts components satisfying them in a way that renders transparent the meaning of their interconnection.

Definition 2 (Contracts). *An assume-guarantee contract $\mathcal{C}$ is a pair of assertions (A, G), called the assumption and the guarantee respectively. The set of environments of $\mathcal{C}$, denoted by $\mathcal{E}_\mathcal{C}$, captures all components E such that*

$$E \leq A \tag{1}$$

In other words, $\mathcal{E}_\mathcal{C} = 2^A$. The set of implementations of $\mathcal{C}$, denoted by $\mathcal{M}_\mathcal{C}$, consists of all components M such that

$$\forall E \in \mathcal{E}_\mathcal{C}. M \oplus E \leq G \tag{2}$$

Example 1. Let $\mathcal{V} = \{x, y\}$ and $\mathcal{C} = (A, G)$ where $A \coloneqq \{\sigma \mid x \in \sigma_i \Leftrightarrow i \mod 2 \neq 0\}$ and $G \coloneqq \{\sigma \mid y \in \sigma_i \Leftrightarrow i \mod 2 = 0\}$. Let $\sigma_1 \coloneqq \langle \{y\}, \{x\}, \varnothing, \{x\}, \varnothing, \ldots \rangle$ and $\sigma_2 \coloneqq \langle \{y\}, \{x\}, \{y\}, \{x\}, \ldots \rangle$. If $E \coloneqq \{\sigma_1, \sigma_2\}$ then $E \in \mathcal{E}_\mathcal{C}$ because $\sigma_1, \sigma_2 \in A$. Let $M_1 \coloneqq \{\sigma_1\}$ and $M_2 \coloneqq \{\sigma_2\}$. Then $M_1 \notin \mathcal{M}_\mathcal{C}$ because $M_1 \oplus E = \{\sigma_1\} \notin G$. However, one can check that $M_2 \in \mathcal{M}_\mathcal{C}$. Note that if $M_3 \coloneqq \varnothing$, then $M_3 \in \mathcal{M}_\mathcal{C}$ as well. The interpretation here is that M_3 satisfies the assume-guarantee semantics of $\mathcal{C}$ vacuously.

Since $2^\mathcal{B}$ is a Boolean algebra (and hence a Heyting algebra [7]), we infer from inequality (2) and the definition of $\oplus$ that $\forall E \in \mathcal{E}_\mathcal{C}. M \leq \neg E \vee G$. Choosing $E = A$ in inequality (1) gives $M \leq \neg A \vee G$. The converse is also true because $\neg E \vee G$ is antitone in E. Thus, we have shown

Proposition 1. *Given $\mathcal{C} = (A, G)$, a component M satisfies $M \in \mathcal{M}_\mathcal{C}$ if and only if*

$$M \leq A \Rightarrow G \tag{3}$$

Proposition 1 characterizes implementations M of $\mathcal{C}$ as those components whose behaviors either do not conform to the behaviors specified in A or are compatible with G. Specifically, it says that $\mathcal{M}_\mathcal{C} = 2^{A \Rightarrow G}$. Since

$$A \Rightarrow (A \Rightarrow G) = \neg A \vee (\neg A \vee G) = \neg A \vee G = A \Rightarrow G, \tag{4}$$

inequalities (1) and (3) yield

Proposition 2. *If $\mathcal{C} = (A, G)$ and $\mathcal{C}^* = (A, A \Rightarrow G)$, then $\mathcal{M}_\mathcal{C} = \mathcal{M}_{\mathcal{C}^*}$ and $\mathcal{E}_\mathcal{C} = \mathcal{E}_{\mathcal{C}^*}$.*

It may be seen from equation (4) why any assume-guarantee contract of the form $\mathcal{C} = (A, A \Rightarrow G)$ is called *saturated*. By the metatheory, we will consider contracts that have the same sets of environments and implementations to be equal, and so by Proposition 2, every contract $\mathcal{C}$ has a unique saturated *canonical* form $\mathcal{C}^*$. This saturated form is often convenient to use when doing contract algebra. In particular, it can also shed light on the meaning of the conjunction operation, which motivates our development of "reactive contracts". To describe the conjunction, we will need the idea of contract refinement.

Definition 3. *We say contract $\mathcal{C}_1 = (A_1, G_1)$ refines a contract $\mathcal{C}_2 = (A_2, G_2)$ and write $\mathcal{C}_1 \preceq \mathcal{C}_2$ if $\mathcal{M}_{\mathcal{C}_1} \subseteq \mathcal{M}_{C_2}$ and $\mathcal{E}_{C_2} \subseteq \mathcal{E}_{C_1}$.*

Clearly, $\preceq$ is a binary relation that is reflexive and transitive and hence is a preorder for the set of all contracts on $\mathcal{B}$. Again, since any two contracts that have the same sets of environments and implementations are considered equal, $\preceq$ is a partial order. The *conjunction* of two contracts $\mathcal{C}_1$ and $\mathcal{C}_2$, denoted by $\mathcal{C}_1 \wedge \mathcal{C}_2$, is a contract that is their largest common lower bound (or meet) with respect to $\preceq$. Formally, $\mathcal{C}_1 \wedge \mathcal{C}_2 \coloneqq \sup\{\mathcal{C} \mid \mathcal{C} \preceq \mathcal{C}_1 \wedge \mathcal{C} \preceq \mathcal{C}_2\}$. For any contract $\mathcal{C}$, we have $\mathcal{E}_\mathcal{C} = 2^A$ and $\mathcal{M}_\mathcal{C} = 2^{A \Rightarrow G}$. Therefore, if $\mathcal{C} \leq \mathcal{C}_1, \mathcal{C}_2$, then by Definition 3, $\mathcal{M}_\mathcal{C} \subseteq \mathcal{M}_{\mathcal{C}_1} \cap \mathcal{M}_{\mathcal{C}_2} = 2^{A_1 \Rightarrow G_1} \cap 2^{A_2 \Rightarrow G_2} = 2^{(A_1 \Rightarrow G_1) \wedge (A_2 \Rightarrow G_2)}$ and $2^{A_1 \vee A_2} = 2^{A_1} \cup 2^{A_2} = \mathcal{E}_{\mathcal{C}_1} \cup \mathcal{E}_{\mathcal{C}_2} \subseteq \mathcal{E}_\mathcal{C}$ (since the intersection/union of powersets of two sets is the powerset of their intersection/union). By the fact that $(A_1 \Rightarrow G_1) \wedge (A_2 \Rightarrow G_2)$ is saturated, we have $\mathcal{C}_1 \wedge \mathcal{C}_2 = (A_1 \vee A_2, (A_1 \Rightarrow G_1) \wedge (A_2 \Rightarrow G_2))$. By induction, one can show the following

Proposition 3. *If for $i = 1, 2, \ldots, n$, C_i are assume-guarantee contracts, then*

$$\wedge_{i=1}^{n} \mathcal{C}_i = (\vee_{i=1}^{n} A_i, \wedge_{i=1}^{n} A_i \Rightarrow G_i) \tag{5}$$

Note that we can apply an analogous argument to the disjunction of contracts (defined as their join) to conclude that the set of all saturated contracts forms a complete lattice. Equation (5) shows that the "parametric contract" formalism given in [15] is exactly the result of applying the conjunction operation to the constituent contracts. That is, if M is an implementation of the conjunction $\wedge_i \mathcal{C}_i$ containing σ such that $\sigma \in E$ where E is an environment of $\wedge_i \mathcal{C}_i$, then there exists at least a $k \in \{1, 2, \ldots, n\}$ such that $\sigma \in A_k$ and for all such k, $\sigma \in G_k$. In other words, for any behavior σ in which the environment satisfies any assumption A_k in $\{A_1, A_2, \ldots, A_n\}$, the system must *react* by providing the corresponding guarantee G_k. Thus in contract conjunctions, 1) the reactions are defined by pairing each A_k with the corresponding G_k and 2) $A_k \Rightarrow G_k$ must hold over the sequence σ in its entirety. The second restriction is partially relaxed in the "dynamic contract" formalism, used for instance in [16], where assumptions are allowed to change over fixed time intervals. Our reactive contract framework will 1) remove the one-to-one restriction to allow for a more flexible assumption-guarantee pairing process, 2) enforces *immediate* guarantee reactions to assumption changes *directly* on each element $\sigma \in \mathcal{B}$, and 3) enables automated synthesis. To motivate this development, we will briefly go over the idea of generative effects.

2.2 Generative effects

The principle of semantic compositionality states that "the meaning of an expression is a function of the meanings of its parts" [22]. One may attempt to interpret this view in the context of system engineering when they replace "expressions" with "systems" and "meanings" with "properties". Let $\mathcal{S}$ be the space of systems, $\oplus : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ be a system-valued infix operator termed *interconnection*, $\mathfrak{C}$ be the set of *all properties*, and $\mathfrak{C}^\mathcal{S} \coloneqq \{\Phi \mid \Phi : \mathcal{S} \to \mathfrak{C}\}$ consisting of functions mapping each system to a property it satisfies. In general, given $\Phi \in \mathfrak{C}^\mathcal{S}$ and $\oplus$, it would be helpful if we can find a binary operation $+$ such that

$\Phi(S_1 \oplus S_2) = \Phi(S_1) + \Phi(S_2)$. Sometimes, for the right classes of systems, properties, and interconnections, such a + operator does, in fact, exist. For example, + can be constructed from a multiplication operation followed by a comparison operation for dynamical systems characterized by the notation of "finite-gain $\mathcal{L}$ stability" that are interconnected by feedback [14]. Sources of failures to capture such a + operator may also arise from to poor choice of Φ, inexact modelling, lack of observability, mechanical fatigue etc. For example, consider a game in which two fair coins c_1 and c_2 are tossed independently and a reward is given every time both come up the same (both heads or both tails) and let Φ be the property: there is a "pattern" for winning. Then if we only observe c_1 or c_2 separately, because the coins are fair, there is no "pattern".

Generative effects are properties resulting from interactions of composed systems that cannot be *explained* by observing individual components [1]. For engineering systems, it makes sense to safeguard against the unpredictability of these effects with *contingency planning* as opposed to simply relying on certain assumptions to hold and blaming the environment when they do not. Our metatheory-compliant formulation of reactive contracts will formalize this idea.

3 Reactive Contracts

3.1 Reactivity

Let $\mathcal{B}$ be as before. For each $\sigma \in \mathcal{B}$ and $i, j \in \mathbb{N} : 0 \leq i < j$, we denote by $\sigma_{i\to j}$ the subsequence of σ spanning from σ_i up to *but not including* σ_j, namely $\langle \sigma_i \rangle_{k=i}^{j-1}$, and by $\sigma_{i\to\infty}$ the infinite sequence $\langle \sigma_k \rangle_{k=i}^{\infty}$. For $A \subseteq \mathcal{B}$ and $k \geq 0$, denote by $\mathrm{Pref}_k(A)$ the set of all prefixes of behaviors in A of length k, $\mathrm{Pref}_k(A) \coloneqq \{\sigma_{0\to k} \mid \sigma \in A\}$ and $\mathrm{Pref}(A) \coloneqq \cup_{k=0}^{\infty} \mathrm{Pref}_k(A)$, the set of all prefixes of A. Let $\cdot$ be the *sequence concatenation* operation mapping from $(\mathrm{Pref}(\mathcal{B}) \times \mathrm{Pref}(\mathcal{B})) \cup (\mathrm{Pref}(\mathcal{B}) \times \mathcal{B})$ to $\mathrm{Pref}(\mathcal{B}) \cup \mathcal{B}$ defined in the obvious way.

Definition 4 (Witness). *Let $\sigma \in \mathcal{B}, A \subseteq \mathcal{B}$ and $i, j \in \mathbb{N}_{\geq 0} \cup \{\infty\} : i < j$. We say that σ is a witness for A from i up until j and write $\sigma \models_{i\to j} A$ if $\sigma_{i\to j} \in \mathit{Pref}(A) \cup A$. If $j \neq \infty$, we consider the witness relation as being strict and write $\sigma \models^s_{i\to j} A$ if $\sigma \models_{i\to j} A$ but $\sigma \not\models_{i\to j+1} A$. If $j = \infty$, the witness relation is always strict.*

To describe and keep track of assumption changes, we appeal to the notion of assigning *signatures* (or labels) to each behavior that undergoes those changes.

Definition 5 (Signature). *Given a set of assertions $\mathcal{A} \subseteq 2^{\mathcal{B}}$, an $\mathcal{A}$-signature is any nonempty sequence $\alpha = \langle \alpha_k \rangle_{k=0}^{m} \in \mathcal{A}^{\infty}$ where $m \in \mathbb{N}_{>0} \cup \{\infty\}$. If $m < \infty$, we say $\sigma \in \mathcal{B}$ is a witness for α and put $\sigma \models \alpha$ if there exists a partitioning sequence $\langle i_k \rangle_{k=0}^{m}$ in $\mathbb{N}_{\geq 0}$ satisfying $0 = i_0 < i_1 < \ldots < i_m$ such that with $i_{m+1} \coloneqq \infty$, we have*

$$\forall k \in \{0, 1, \ldots, m\}. \sigma \models_{i_k \to i_{k+1}} \alpha_k. \tag{6}$$

We say that σ is a strict witness for the signature α and write $\sigma \models^s \alpha$ if the witness relation in equation (6) *is strict. Analogously, if $m = \infty$, then $\sigma \models \alpha$ if there exists a (strictly monotone) partitioning sequence $\langle i_k \rangle_{k=0}^{\infty}$ in $\mathbb{N}_{\geq 0}$ satisfying $i_0 = 0$ and equation* (6) *with $\{0, 1, \ldots, m\}$ replaced by $\mathbb{N}_{\geq 0}$.*

In general, a given $\sigma \in \mathcal{B}$ may be a strict witness for more than one signature in $\mathcal{A}^\infty$. For example, if $\mathcal{A} = \{A_1, A_2\}$ where $A_1 \cap A_2 \neq \varnothing$, then any behavior $\sigma \in A_1 \cap A_2$ satisfies $\sigma \models^s \langle A_j \rangle$ for $j \in \{1, 2\}$. This may still be the case even when $A_1 \cap A_2 = \varnothing$. For $\mathcal{V} = \{x, y\}$. $A_1 = \{\{x\}\}^\omega$ and $A_2 = \{\{y\}\}^\omega \cup \{\sigma\}$ where σ satisfies $\sigma_k = \{x\}$ for $k = 0$ and $\sigma_k = \{y\}$, otherwise. Then $\sigma \models^s \langle A_1, A_2 \rangle$ and $\sigma \models^s \langle A_2 \rangle$. This non-uniqueness makes it unclear as to which assumption change sequence should be considered and how/when to properly react to it. Being able to restrict the set of assumptions so that this does not happen is necessary because in order to react at all, the system must be able to know which assumption to operate under next. The following proposition gives a sufficient condition.

Proposition 4. *Let $\mathcal{A}$ be a collection of assertions, then*

$$\forall \alpha \in \mathcal{A}^\infty . \forall \sigma \in \mathcal{B} . \sigma \models^s \alpha \Rightarrow (\forall \beta \in \mathcal{A}^\infty . \sigma \models^s \beta \Rightarrow \beta = \alpha) \tag{7}$$

if

$$\forall A_1, A_2 \in \mathcal{A} . A_1 \neq A_2 \Rightarrow \mathit{Pref}_1(A_1) \cap \mathit{Pref}_1(A_2) = \varnothing. \tag{8}$$

Proof. Assume that $\mathcal{A}$ satisfies (8). Let $\alpha, \beta \in \mathcal{A}^\infty$ and $\sigma \in \mathcal{B}$ be such that $\sigma \models^s \alpha$ and $\sigma \models^s \beta$. Let $m \in \mathbb{N} \cup \{\infty\}$ be the length of α. By the fact that $\sigma_0 \in \mathrm{Pref}_1(\alpha_0) \cap \mathrm{Pref}_1(\beta_0)$ and $\forall A \in \mathcal{A} . A \neq \alpha_0 \Rightarrow \mathrm{Pref}_1(A) \cap \mathrm{Pref}_1(\alpha_0) = \varnothing$, we conclude $\alpha_0 = \beta_0$. Suppose that up to $n < m$, $\alpha_k = \beta_k$ for all k satisfying $0 \leq k \leq n$. We will show that α_{n+1} and β_{n+1} are defined and equal to one another. Indeed, since $n < m$, the $(n+1)^{\text{th}}$ term of α, α_{n+1}, exists. Let $\langle i_k \rangle_{k=0}^{m}$ be a partitioning sequence for $\sigma \models^s \alpha$ given by Definition 5. By strictness and the induction hypothesis, we have $\sigma_{i_n \to (i_{n+1}+1)} \not\models \alpha_n = \beta_n$. Since $\sigma \models^s \beta$, it follows that β_{n+1} exists as well. From $\sigma \models^s \alpha$, if $n + 2 \leq m$, we have $\sigma \models^s_{i_{n+1} \to i_{n+2}} \alpha_{n+1}$ and, in particular, $\sigma_{i_{n+1}} \in \mathrm{Pref}_1(\alpha_{n+1}) \cap \mathrm{Pref}_1(\beta_{n+1})$, which by (8) yields $\alpha_{n+1} = \beta_{n+1}$. If $n + 2 > m$, then $\sigma \models^s_{i_{n+1} \to \infty} \alpha_{n+1}$, arguing similarly, we arrive at the additional conclusion that β also has length m. This implies (7). □

Any $\mathcal{A}$ that satisfies (8) is called *initially disjoint*. Hence, Proposition 4 says that if $\mathcal{A}$ is a set of assertions that are initially disjoint then any behavior is a strict witness for at most one $\mathcal{A}$-signature. Let $\mathcal{B}_\mathcal{A} \coloneqq \{\sigma \in \mathcal{B} \mid \exists \alpha \in \mathcal{A}^\infty . \sigma \models^s \alpha\}$, the set of behaviors that have $\mathcal{A}$-signatures. If $\mathcal{A}$ is initially disjoint, then the function $\mathcal{U}_\mathcal{A} : \mathcal{B}_\mathcal{A} \to \mathcal{A}^\infty$ mapping each $\sigma \in \mathcal{B}_\mathcal{A}$ to the unique signature $\mathcal{U}_\mathcal{A}(\sigma) \in \mathcal{A}^\infty$ for which it is a witness is well-defined. Finally, for any $M \subseteq \mathcal{B}_\mathcal{A}$, we denote by $\mathcal{U}_\mathcal{A}(M)$ the set of signatures generated by M, namely, $\{\mathcal{U}_\mathcal{A}(\sigma) \mid \sigma \in M\}$.

3.2 Contracts

Definition 6 (Reactive contracts). *A reactive assume-guarantee contract $\mathcal{C}$ is a 4-tuple $(\mathcal{A}, \mathcal{G}, \Delta, R)$ such that*

1. *$\mathcal{A}, \mathcal{G} \subseteq 2^{\mathcal{B}}$ are called the assumption and guarantee sets respectively. $\mathcal{A}$ is required to be initially disjoint.*
2. *$\Delta \subseteq \mathcal{A}^\infty$ is called the contingency set, consisting of assumption change scenarios that may happen.*
3. *$R \subseteq (\mathcal{A} \times \mathcal{G})^\infty$ is called the reaction set.*

Observe that $\mathcal{A}$ and $\mathcal{G}$ are not necessarily of the same cardinality and that from each $r \in R$, we can obtain an unique $\mathcal{A}$-signature by "projecting away the $\mathcal{G}$ dimension". We denote the projection function by $\Pi_{\mathcal{A}} : R \to \mathcal{A}^\infty$ so that $\Pi_{\mathcal{A}}(\langle A_k, G_k \rangle_{k=0}^m) \coloneqq \langle A_k \rangle_{k=0}^m$ for any $\langle A_k, G_k \rangle_{k=0}^m \in R$.

Definition 7 (Environment). *An environment for a reactive contract $\mathcal{C} = (\mathcal{A}, \mathcal{G}, \Delta, R)$ is any $E \subseteq \mathcal{B}_{\mathcal{A}}$ such that $\mathcal{U}_{\mathcal{A}}(E) \subseteq \Delta$, namely each $\sigma \in E$ is a strict witness for some $\mathcal{A}$-signature in Δ.*

Thus, for a reactive contract, assumptions about its environment's behaviors are allowed to change according to the contingency specified in Δ. As these assumptions change, the system should provide the corresponding guarantees as specified by the reaction set R. We characterize R via the following definitions.

Definition 8 (Reactive satisfaction). *Let $\sigma \in \mathcal{B}$, $r = \langle (A_k, G_k) \rangle_{k=0}^m \in R$, we say that σ reactively satisfies r and write $\sigma \models^\rho r$ if the following hold*

1. *$\sigma \models^s \Pi_{\mathcal{A}}(r)$ with the partitioning sequence $\langle i_k \rangle_{k=0}^m$.*
2. *(a) If $m < \infty$, then $\forall k \in \{0, 1, \ldots, m\}. \sigma_{i_k \to i_{k+1}} \models G_k$ with $i_{m+1} \coloneqq \infty$.*
   (b) otherwise, $\forall k \in \mathbb{N}_{\geq 0}. \sigma_{i_k \to i_{k+1}} \models G_k$.

Definition 9 (Implementation). *An implementation of a reactive contract $\mathcal{C} = (\mathcal{A}, \mathcal{G}, \Delta, R)$ is any $M \subseteq \mathcal{B}$ such that for any environment E of $\mathcal{C}$, we have*

$$\forall \sigma \in (M \cap E). \exists r \in R. \sigma \models^\rho r \wedge \Pi_{\mathcal{A}}(r) = \mathcal{U}_{\mathcal{A}}(\sigma).$$

Intuitively, an implementation consists of all behaviors σ in which either the assumptions do not change according to Δ, i.e., $\mathcal{U}_{\mathcal{A}}(\sigma) \notin \Delta$, or the system reacts according to instructions specified by the set R, namely there exists a reaction $r \in R$, such that σ reactively satisfies r, in which the system tries its best to satisfy the guarantee corresponding to the current assumption for as long as the latter holds and is required to immediately adapt to any new assumption and commit itself to the new obligation. Let us compare this machinery to "standard" assume-guarantee contracts. First, we mention that the following holds.

Proposition 5. *Corresponding to each any standard assume-guarantee contract $\mathcal{C} = (A, G)$ is a reactive assume-guarantee contract $\mathcal{C}_r = (\mathcal{A}, \mathcal{G}, \Delta, R)$ with $\mathcal{A} = \{A\}, \mathcal{G} = \{G\}, \Delta = \{\langle A \rangle\}$ and $R = \{\langle (A, G) \rangle\}$ such that $\mathcal{C} = \mathcal{C}_r$ in the sense that they have same sets of environments and implementations.*

Recall that any parametric assume-guarantee contract is simply the conjunction of a set of "standard" assume-guarantee contracts (which is again a standard assume-guarantee contract) by Proposition 5, there is a reactive version for it. In particular, when all assumptions are initially disjoint, we have the following the generalization of Proposition 5.

Proposition 6. *If $n \geq 1$, $\{A_1, A_2, \ldots, A_n\}$ is a set of initially disjoint assertions and for $i \in \{1, 2, \ldots, n\}$, $\mathcal{C}_i = (A_i, G_i)$ are assume-guarantee contracts, then there exists a reactive assume-guarantee contract $\mathcal{C}_r$ such that $\wedge_{i=1}^{n} \mathcal{C} = \mathcal{C}_r$.*

Proof. Let $\mathcal{C}_r = (\mathcal{A}, \mathcal{G}, \Delta, R)$ be defined with

- $\mathcal{A} \coloneqq \{A_1, A_2, \ldots, A_n\}$
- $\mathcal{G} \coloneqq \{G_1, G_2, \ldots, G_n\}$
- $\Delta \coloneqq \{\langle A_1 \rangle, \langle A_2 \rangle, \ldots, \langle A_n \rangle\}$
- $R \coloneqq \{\langle (A_1, G_1) \rangle, \langle (A_2, G_2) \rangle, \ldots, \langle (A_n, G_n) \rangle\}$

Then, $E \in \mathcal{E}_{\mathcal{C}}$ if and only if

$$\begin{aligned} &\Leftrightarrow && \forall \sigma \in E. \sigma \in \vee_{i=1}^{n} A_i \\ &\Leftrightarrow && \forall \sigma \in E. \exists i \in \{1, 2, \ldots, n\}. \sigma \in A_i \\ &\Leftrightarrow && \forall \sigma \in E. \exists i \in \{1, 2, \ldots, n\}. \mathcal{U}_{\mathcal{A}}(\sigma) = \langle A_i \rangle \subseteq \Delta \end{aligned}$$

which holds if and only if $E \in \mathcal{E}_{\mathcal{C}_r}$. Also, $M \in \mathcal{M}_{\mathcal{C}} \Leftrightarrow \forall \sigma \in M. \sigma \in \wedge_{i=1}^{n} (A_i \Rightarrow G_i)$. Since the A_i's are initially disjoint, and therefore disjoint, there are two cases: either $\sigma \in \wedge_{i=1}^{n} \neg A_i$, in which case $\mathcal{U}_{\mathcal{A}}(\sigma) \notin \Delta$, or there is an A_i such that $\sigma \in A_i \wedge G_i$, in which case, $\sigma \models^{\rho} \langle (A_i, G_i) \rangle$ and $\mathcal{U}_{\mathcal{A}}(\sigma) = \Pi_{\mathcal{A}}(\langle (A_i, G_i) \rangle) = \langle A_i \rangle$. This implies $M \in \mathcal{M}_{\mathcal{C}_r}$. On the other hand, $M \in \mathcal{M}_{\mathcal{C}_r}$ implies $\forall \sigma \in M$, either $\sigma \models^{\rho} \langle (A_i, G_i) \rangle$ for some $i \in \{1, 2, \ldots, n\}$, which by Definition 8, implies that $\sigma \in A_i \wedge G_i$, or $\mathcal{U}_{\mathcal{A}}(\sigma) \notin \Delta$, which implies that $\sigma \in \wedge_{i=1}^{n} \neg A_i$. □

The following example shows the greater flexibility offered by reactive contracts over parametric ones.

Example 2. Let A_1, A_2 be initially disjoint and $\mathcal{C} = (A_1 \vee A_2, (A_1 \Rightarrow G_1) \wedge (A_2 \Rightarrow G_2)$ and $\tilde{\mathcal{C}}_r = (\tilde{\mathcal{A}}, \tilde{\mathcal{G}}, \tilde{\Delta}, \tilde{R})$ where $\tilde{\mathcal{A}} = \{A_1, A_2\}$, $\tilde{\mathcal{G}} = \{G_1, G_2\}$, $\tilde{\Delta} = \{\langle A_1 \rangle, \langle A_2 \rangle, \langle A_1, A_2 \rangle\}$, $\tilde{R} = \{\langle (A_1, G_1) \rangle, \langle (A_2, G_2) \rangle, \langle (A_1, G_1), (A_2, G_2) \rangle\}$. We can see $\tilde{\mathcal{C}}_r \preceq \mathcal{C}$ from the fact that by Proposition 6, $\mathcal{C} = \mathcal{C}_r = (\mathcal{A}, \mathcal{G}, \Delta, R)$ where $A = \tilde{\mathcal{A}}$, $G = \tilde{\mathcal{G}}$, $\Delta = \{\langle A_1 \rangle, \langle A_2 \rangle\}$, and $R = \{\langle (A_1, G_1) \rangle, \langle (A_2, G_2) \rangle\}$. With the inclusion of $\langle (A_1, G_1), (A_2, G_2) \rangle$ in $\tilde{R}$, $\tilde{\mathcal{C}}_r$ is receptive to environments whose behaviors exhibit a change in assumptions from A_1 to A_2 and requires implementations to adapt accordingly by changing their guarantee from G_1 to G_2. On the other hand, $\mathcal{C}_r$ only specifies the set of implementations to be those behaviors in which either neither A_1 or A_2 is satisfied or at least a pair (A_i, G_i) is always satisfied.

In a similar manner to standard assume-guarantee contracts, we can talk about the (saturated) canonical form of reactive assume-guarantee contracts. First, let $\mathfrak{R} \triangleq (\mathcal{A} \times \mathcal{G})^{\infty}$, the set of all $(\mathcal{A} \times \mathcal{G})$-signatures and $\Delta \Rightarrow R \coloneqq \{r \mid r \in \mathfrak{R} \wedge \Pi_{\mathcal{A}}(r) \notin \Delta\} \cup R$, the set of all $(\mathcal{A} \times \mathcal{G})$-signatures that are either a reaction in R or have an assumption change sequence not specified in the contingency Δ. The proof of the following proposition is given in the appendix.

Proposition 7. *If $\mathcal{C} = (\mathcal{A}, \mathcal{G}, \Delta, R)$ is a reactive contract, then the canonical form of $\mathcal{C}$ defined as $\mathcal{C}^{\star} \coloneqq (\mathcal{A}, \mathcal{G}, \Delta, \Delta \Rightarrow R)$ satisfies $\mathcal{C}^{\star} = \mathcal{C}$.*

One may ask what the conjunction is for reactive contracts. The following proposition, proved in the appendix, provides an answer to that question

Proposition 8. *Let there be canonical reactive contracts $\mathcal{C}_1$, $\mathcal{C}_2$ such that $\mathcal{C}_1 = (\mathcal{A}_1, \mathcal{G}_2, \Delta_1, R_1)$, $\mathcal{C}_2 = (\mathcal{A}_2, \mathcal{G}_2, \Delta_2, R_2)$ and $\mathcal{A}_1 \cup \mathcal{A}_2$ is initially disjoint, then the metatheoretic conjunction is $\mathcal{C}_1 \wedge \mathcal{C}_2 = (\mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{G}_1 \cap \mathcal{G}_2, \Delta_1 \cup \Delta_2, R_1 \cap R_2)$.*

Now we will move on to show how this formalism can be applied reactive synthesis.

4 Reactive Contracts for GR(1) Games

Reactive synthesis refers to the automatic correct-by-construction synthesis of a reactive system from formal specifications. Linear temporal logic (LTL) is a language whose formulae are built from a finite set of logical (e.g, $\neg, \wedge, \vee$), temporal (e.g., $\Box, \Diamond$ for "always" and "eventually") operators and atomic propositions [9]. An LTL formula can be used to check a system trace for satisfaction of properties such as "always eventually `atom_prop` will be true" ($\Box\Diamond$`atom_prop`) or "`atom_prop` must always hold" ($\Box$`atom_prop`). In general, synthesis from LTL specifications is 2EXPTIME-complete [26]. Fortunately, an expressive subset of LTL has been identified and shown to allow for relatively efficient synthesis algorithms with cubic time complexity [19]. This subset is called General Reactivity of Rank 1 (GR(1)) and offers the following specification format:

$$\underbrace{(\theta^e \wedge \Box\rho^e \wedge \wedge_{0\leq i\leq m}\Box\Diamond J_i^e)}_{A} \Rightarrow \underbrace{(\theta^s \wedge \Box\rho^s \wedge \wedge_{0\leq i\leq n}\Box\Diamond J_i^s)}_{G} \tag{9}$$

The formula in (9) is to be interpreted on a 2-player turn-based game between an environment e and a system (implementation) s over a set of variables $\mathcal{V}$ that can be decomposed as $\mathcal{V}^e \cup \mathcal{V}^s$. A round of the game consists of two turns with the first being taken by the environment to set values for variables in $\mathcal{V}^e$ and the second taken by the system to set values for variables in $\mathcal{V}^s$. By the end of each round, all variables in $\mathcal{V}$ will have been assigned a value. Such a valuation of $\mathcal{V}$ defines a game *state*. A *play* is an infinite sequence of states. In (9), for $p \in \{e, s\}$, θ^p is a constraint over the set of initial states, ρ^p is a safety constraint over the current and the next states, and J_i^p is a liveness constraint specifying a set of states that are required to always eventually be visited. Without loss of generality, we can assume the variables in $\mathcal{V}$ are Boolean, in which case the contract formalism discussed earlier captures (9) exactly. That is, the set $\mathcal{B} = (2^{\mathcal{V}})^\omega$ consists of all possible plays of the game and (9) is a *GR(1) contract* of the form $(A, A \Rightarrow G)$ for which the powerset of the set of all plays in which the antecedent A holds is the set of all environments and the powerset of the set of all plays in which the antecedent A fails or the consequent G holds is the set of all implementations. A GR(1) contract is said to be *realizable* if the system has a strategy such that no matter what the environment does, the resulting plays are implementations of the contract. Note that the number 1 in "GR(1)" refers to the fact that the

underlying liveness implication $\wedge_{0\leq i\leq m}\Box\Diamond J_i^e \Rightarrow \wedge_{0\leq i\leq n}\Box\Diamond J_i^s$ is considered to be 1 Streett pair. Observe, therefore, that the set of GR(1) contracts is not closed with respect to the contract algebra defined above. In fact, taking the conjunction of k GR(1) contracts in the sense of the metatheory can result in a single GR(k) contract (defined as having a conjunction of k Streett pairs [25]) instead.

Being of the form $(A, A \Rightarrow G)$, GR(1) contracts are certainly not reactive: as mentioned, a play in which the system successfully incapacitates the environment from fulfilling its obligations in the antecedent is considered to be an implementation. This vacuous satisfaction may result in unintended behaviors: in fact, there has recently been work aimed at finding "environmentally-friendly" implementations that allow the environment to satisfy its promises [18]. Of course, a designer may attempt to mitigate this problem by making the contract reactive, to which end, they will need to somehow come up with a proper characterization of the sets $\mathcal{A}$, $\mathcal{G}$, Δ, and R. Relatively speaking, the one that is often the "easiest" to characterize among these is $\mathcal{G}$ because system designers usually have some idea of a set of basic services the system should provide. In addition, they can also think of a set of less desirable but still acceptable services that the system may resort to when a failure occurs. The set $\mathcal{A}$ can be less straight forward to construct because it is difficult to predict failures by the very definition of generative effects. In addition, the multiple possible configurations (some may be unreachable) of the system/environment at which a failure happens often determine the possibility of recovery and therefore should be considered. For many such reasons, system designers may only come up with a set of fragmentary, perhaps very specific and incomplete assumptions and guarantees that they think may be relevant. In so far as $\mathcal{A}$ and $\mathcal{G}$ are fragmentary and the relationships between their elements are unknown, one is not quite ready to specify Δ and R. The contract synthesis question becomes: how do we synthesize from these sets of *coarse* assumptions and guarantees a compact and fundamental road map for specifying reactivity?

We observe that a key relation between elements of $\mathcal{A}$ and $\mathcal{G}$ is that of realizability since after all including an unrealizable pair in a reaction from R does not guarantee an implementation. More importantly, this relation can be efficiently computed for GR(1) games as part of the same synthesis algorithms mentioned earlier. From this point on, we will denote by $\mathcal{R} \subseteq \mathcal{A}\times\mathcal{G}$ the *realizability relation* that satisfies $(A, G) \in \mathcal{R}$ if and only if (A, G) is realizable. From $\mathcal{R}$, an answer to the above question may be found in the form of a Galois connection between $2^{\mathcal{A}}$ and $2^{\mathcal{G}}$.

Definition 10 (Galois connection). *Given two partially ordered sets $(A, \leq_A)$ and $(G, \leq_G)$, an (antitone) Galois connection between these two sets consists of a pair of maps $^* : A \to G$ and $_* : G \to A$ such that for any $a \in A$ and $g \in G$*

$$a \leq_A g_* \Leftrightarrow g \leq_G a^*$$

where a^* and g_* denote the applications of * and $_*$ to $a \in A$ and $g \in G$.

Define the "forward" map * by $^* : S_{\mathcal{A}} \mapsto \{G \mid \forall A \in S_{\mathcal{A}}.(A, G) \in \mathcal{R}\}$ and the "pullback" map $_*$ by $_* : S_{\mathcal{G}} \mapsto \{A \mid \forall G \in S_{\mathcal{G}}.(A, G) \in \mathcal{R}\}$. One can verify that these maps satisfy the requirement of Definition 10 and therefore form a Galois connection. It is well know that the compositions of Galois connection operators, namely, ${}_*\circ^*$ and ${}^*\circ_*$ are dual closure operators and give rise to fixpoint pairs that form isomorphic complete lattices when ordered by set inclusion [11]. In our case, the one that comes from the coarse assumptions will be called the *assumption lattice* and the other the *guarantee lattice*.

Each of these fixpoint pairs is of the form $(S_{\mathcal{A}}, S_{\mathcal{G}})$ where $S_{\mathcal{A}}$ is the largest subset of $\mathcal{A}$, each element of which is realizable with all guarantees in the subset $S_{\mathcal{G}}$ of $\mathcal{G}$ and vice versa. Observe that since each assumption in $S_{\mathcal{A}}$ is realizable with all guarantees in $S_{\mathcal{G}}$, we can take the disjunction of all fragmentary assumptions in $S_{\mathcal{A}}$ to form a single new assumption that is also realizable with any guarantee from $S_{\mathcal{G}}$. Dually, when actions taken in the GR(1) games are reversible (hence the ability to satisfy two liveness goals separately implies the ability to satisfy them jointly), we can get a single guarantee from $S_{\mathcal{G}}$ by taking the conjunction of all its guarantees. In fact, we can get a smaller representation of the fixpoints with some preprocessing. We can simplify $S_{\mathcal{G}}$ ($S_{\mathcal{A}}$) to $S'_{\mathcal{G}}$ ($S'_{\mathcal{A}}$) by subtracting from it all guarantees (assumptions) contained in any descendant (ancestor) of the node corresponding to $S_{\mathcal{G}}$ ($S_{\mathcal{A}}$) on the guarantee (assumption) lattice. Note that operation may result in an empty set (and if that is the case, we will denote it by the symbol "$\wedge$" ("$\vee$") as in Fig. 2). Thus, these new and compact fixpoints $(S'_{\mathcal{A}}, S'_{\mathcal{G}})$ allow us to define new sets of assumptions and guarantees $\mathcal{A}'$ and $\mathcal{G}'$ with complete lattice structures that can be used to specify Δ and R. On the assumption lattice, the greater elements represent assumptions that are more favorable to the system and conversely on the guarantee lattice, the greater elements are more "difficult" for the system to satisfy (see Fig. 2). In other words, descending any chain from the assumption lattice is equivalent to experiencing more and more restrictive assumptions that may be associated with more severe failures. A chain on the guarantee lattice however represents all possible guarantees that can be realizable if the largest element of the chain is realizable.

Note that the potentially most costly step in the described procedure is the computation of the realizability relation $\mathcal{R}$ even though this should often be done offline during the design/planning process. It is, nevertheless, easy to parallelize and also optimize this computation using a priori knowledge about $\mathcal{A}$ and $\mathcal{G}$ (to be discussed in the next section). In addition, full knowledge of $\mathcal{R}$ may not be necessarily for most applications and for that reason it can be incrementally refined. The following case study will demonstrate this method.

5 Case Study: a Reactive GR(1) Contract on 3 Islands

Fig. 1 contains two screenshots from a simulation in [23] that describes a reactive GR(1) game involving two land robots that navigate and manipulate an uncertain gridworld environment consisting of 3 islands connected by 2 bridges

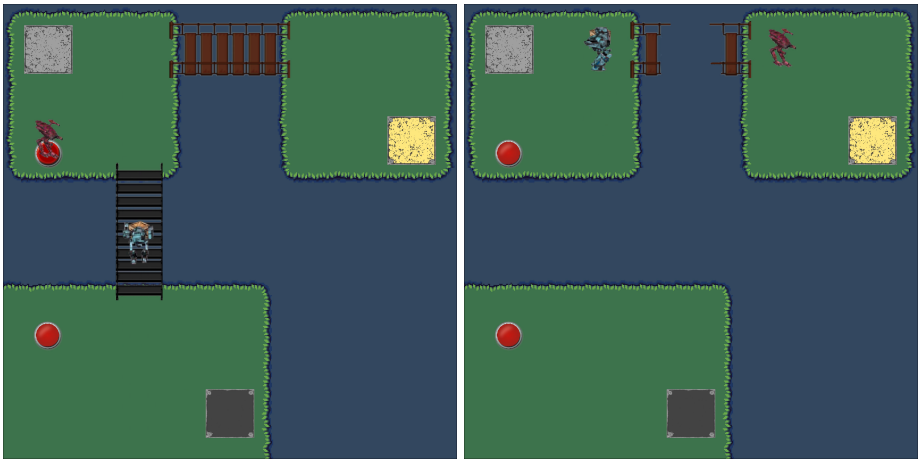

Fig. 1. Two screenshots from an implementation of a reactive GR(1) contract. The simulation and synthesis code is available at [23].

in order to satisfy some liveness objectives. In the left screenshot, the "transporter" robot (blue) is shown carrying a crate across the black bridge, which only deploys when at least one of the red buttons (which may unexpectedly fail) is held down by either robot (in this scene, the "supervisor" robot). The brown bridge is always available but, like the buttons, can suddenly go out of service (see right screenshot). The crate (movable only by the transporter robot) can only be picked up from the black square patch (a factory) and deposited at either the silver or yellow square patches (shipping ports), after which it will respawn at the black patch. The set of coarse guarantees $\mathcal{G}$ consists of 7 elements:

- `box_r2_far` (resp., `box_r2_near`): the supervisor robot stands on the yellow (resp., silver) square patch to supervise the dropping off of the crate there infinitely often.
- `box_far` (resp., `box_near`): the crate gets dropped off at the yellow (resp., silver) square patch infinitely often (possibly without the supervisor robot being present).
- `r2_far` (resp., `r2_near`, `r2_home`): the supervisor robot patrols the yellow (resp., silver, black) square patch infinitely often.

The set $\mathcal{A}$ that coarsely characterizes possible operating scenarios is constructed in an semi-automatic manner by taking the conjunction of all variables appearing in elements of the product set obtained from the sets $\{\texttt{r1\_home}, \texttt{r1\_near}, \texttt{r1\_far}\}$, $\{\texttt{r2\_home}, \texttt{r2\_near}, \texttt{r2\_far}\}$, and $2^{\{\texttt{bridge},\texttt{button1},\texttt{button2}\}}$. An element of $\mathcal{A}$ can be $\texttt{r1\_home} \wedge \texttt{r2\_far} \wedge \texttt{bridge} \wedge \texttt{button1}$, denoting the condition that the transporter robot r1 is on the "home" island (the one with the black patch), the supervisor robot r2 is on the "far" island (yellow patch) and the button on the island with the silver patch, button2, is broken. We observe that $\mathcal{A}$ has 72 elements and

is initially disjoint. The Galois connection calculation will reduce this relatively large number of cases to only 9 in $\mathcal{A}'$ as can be seen from Fig. 2.

A brute-force calculation of $\mathcal{R}$ on $\mathcal{A} \times \mathcal{G}$ (which has 72×7 elements) using `slugs`[12] on a laptop with an Intel i7-4720HQ processor and 16GB of RAM took approximately 40 minutes. It took about 3 seconds to generate the lattices using a fixpoint calculation algorithm from [21]. Some of these calculations for $\mathcal{R}$ are redundant and can be inferred from the others: for example, if a guarantee is not realizable from an assumption, then it will not be realizable from the same assumption conjoined with a new failure. Note also that since all actions performed by the two robots in this example are reversible, if under an assumption, two liveness guarantees can be realized separately, their conjunction will also be realizable. This observation will also be used to synthesize one of the most basic forms of reactive contracts, namely: an "optimal" reactive contract, in which the reaction R is defined to consist of all finite and infinite sequences whose elements are of the form (α, γ) where α is an assumption from a node on the assumption lattice and γ is the conjunction of the corresponding guarantee on the guarantee lattice together with all guarantees contained in its descendant nodes. The contigency Δ is defined to be the sequences obtained by projecting away γ from any sequence in R. In Fig. 2, if $\alpha = \texttt{bridge} \wedge (\texttt{r2\_far} \vee \texttt{r2\_near})$ then $\gamma = \square\lozenge\texttt{r2\_near} \wedge \square\lozenge\texttt{r2\_far}$.

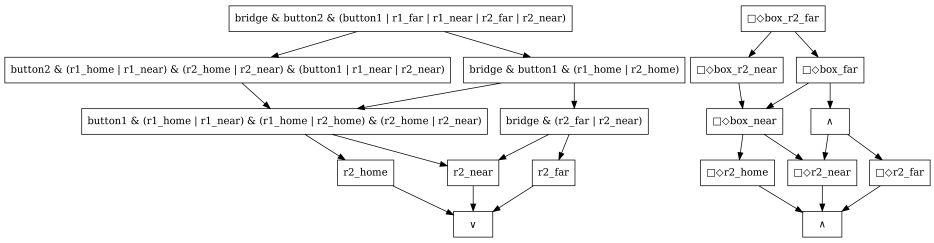

Fig. 2. Order isomorphic assumption (left) and guarantee (right) lattices of Galois fixpoints computed from coarse assumptions and guarantees. The bottom element on the assumption lattice corresponds to the worst case scenario where any assumption can hold and the one on the guarantee lattice corresponds to "no guarantee".

Consider a scenario in which we are given the lattices $\mathcal{A}'$ and $\mathcal{G}'$ and a set $F \subseteq \mathfrak{F}$ where $\mathfrak{F}$ consists of Boolean variables encoding all impending failures (in the 3 island scenarios, $\mathfrak{F} = \{\texttt{button1}, \texttt{button2}, \texttt{bridge}\}$) along with a set $g_{\min} \subseteq \mathcal{G}'$ consisting of baseline guarantees that we would like the system to maintain. Algorithm 1 shows that we can use the assume-guarantee lattices to find an implementation that is robust against F while maintaining at least $g_{\min}$ starting from an initial configuration A_{init}. In line 2 of the algorithm, a check is performed to see if the initial configuration A_{init} is a valid assumption, if not an error will be thrown (line 3). In line 5, the algorithm finds the least node on $\mathcal{G}'$ such that γ and its descendants contains $g_{\min}$ (this always exists and is unique

Algorithm 1 Find robust reactive GR(1) implementation for prioritized failures

1: **function** FINDROBUSTIMPLEMENTATION($\mathcal{A}', \mathcal{G}', A_{\text{init}}, F, g_{\min}$)
2: **if** $A_{\text{init}} \notin \cup_{\alpha \in \mathcal{A}'} \alpha$ **then**
3: error out: "initial state is not in contract!"
4: **else**
5: $\gamma \leftarrow \sup_{\leq_{\mathcal{G}'}} g_{\min}$
6: $\alpha \leftarrow$ the $\mathcal{A}'$ node corresponding to γ
7: $\alpha_{\text{inv}} \leftarrow \alpha \vee (\vee_{a \in \text{ancestors}(\alpha)} a)$ with variables in F replaced by `False` and variables in $\mathfrak{F} \setminus F$ replaced by `True`
8: **if** ISREALIZABLE($A_{\text{init}}, \gamma \wedge \alpha_{\text{inv}}$) **then**
9: $\gamma' \leftarrow$ largest guarantee such that ISREALIZABLE($A_{\text{init}}, \gamma' \wedge \alpha_{\text{inv}}$)
10: **return** GETSTRATEGY($A_{\text{init}}, \gamma' \wedge \alpha_{\text{inv}}$)
11: **else**
12: error out: "robust strategy doesn't exist!"

by the completeness of $\mathcal{G}'$). Line 6 defines α as the corresponding node to γ on $\mathcal{A}'$. Hence, $\alpha \vee (\vee_{a \in \text{ancestors}(\alpha)} a)$ corresponds to the most relaxed assumption for which $g_{\min}$ can be guaranteed. In other words, all configurations from which $g_{\min}$ can be satisfied are contained in it. In line 7, an invariant constraint α_{inv} is computed from $\alpha \vee (\vee_{a \in \text{ancestors}(\alpha)} a)$ by assuming all failures in F have occurred. This represents configurations from which γ can still be guaranteed. In line 8, a check is performed to see if it is possible to satisfy γ while maintaining α_{inv}. If the answer is yes, the algorithm attempts to find a better guarantee than γ (possibly using bisection) and will return a robust strategy that guarantees it (lines $9-10$), otherwise it will throw an error saying that such a robust strategy does not exist (line 12).

In the 3 island example, if we start out at an initial configuration where at least one robot is not on the "home" island (the one with the black patch), $g_{\min}$ is anything, and $F = \{\texttt{button1}\}$ then algorithm 1 will return a robust strategy with $\gamma' = \mathcal{G}'$ (all guarantees) which never allows both robots to be on the "home" island at the same time. In particular, if $g_{\min} = \{\texttt{box\_r2\_far}\}$ then α_{inv} will be equal to `r1_far | r1_near | r2_far | r2_near`.

6 Conclusion and Future Work

In this work, we have developed a metatheory-compliant contract framework that focuses on specifying a system's reactions to antecedent failures and related it to a reactive synthesis setting involving GR(1) contracts. We have also looked at automating the process of synthesizing and simplifying reactive contracts by computing fixpoint pairs of a certain Galois connection between the system's assumption and guarantee sets and carried out a simulated case study that concretizes our ideas. In the future, we will look into 1) exploring and developing methods to automate the process of finding coarse assumptions with [2] as a possible starting point 2) studying compositions of systems specified by reactive contracts as well as the propagation of failures through these compositions.

References

1. Elie M. Adam. *Systems, generativity and interactional effects*. PhD thesis, Massachusetts Institute of Technology, 2017.
2. Rajeev Alur, Salar Moarref, and Ufuk Topcu. Counter-strategy guided refinement of gr (1) temporal logic specifications. In *2013 Formal Methods in Computer-Aided Design*, pages 26–33. IEEE, 2013.
3. André Arnold and Damian Niwinski. *Rudiments of calculus*, volume 146. Elsevier, 2001.
4. Shaun Azzopardi, Gordon J. Pace, and Fernando Schapachnik. Contract automata with reparations. In *JURIX*, volume 271 of *Frontiers in Artificial Intelligence and Applications*, pages 49–54. IOS Press, 2014.
5. Albert Benveniste, Benoît Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, Philipp Reinkemeier, Alberto Sangiovanni-Vincentelli, Werner Damm, Tom Henzinger, and Kim G. Larsen. Contracts for Systems Design: Theory. Research Report RR-8759, Inria Rennes Bretagne Atlantique ; INRIA, July 2015.
6. Roderick Bloem. Reactive synthesis. In *2015 Formal Methods in Computer-Aided Design (FMCAD)*, pages 3–3, Sep. 2015.
7. Stanley Burris and H.P. Sankappanavar. *A course in universal algebra*. Graduate texts in mathematics. Springer-Verlag, 1981.
8. Andrea Censi. A mathematical theory of co-design. Technical report, Laboratory for Information and Decision Systems, MIT, September 2016. Submitted and conditionally accepted to IEEE Transactions on Robotics.
9. Edmund M. Clarke Jr., Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. *Model checking*. MIT press, 2018.
10. Sumanth Dathathri, Scott C. Livingston, and Richard M. Murray. Enhancing tolerance to unexpected jumps in GR(1) games. In Sonia Martínez, Eduardo Tovar, Chris Gill, and Bruno Sinopoli, editors, *Proceedings of the 8th International Conference on Cyber-Physical Systems, ICCPS 2017, Pittsburgh, Pennsylvania, USA, April 18-20, 2017*, pages 37–47. ACM, 2017.
11. Klaus Denecke, Marcel Erné, and Shelly L. Wismath. *Galois connections and applications*, volume 565. Springer Science & Business Media, 2013.
12. Rüdiger Ehlers and Vasumathi Raman. Slugs: Extensible gr (1) synthesis. In *International Conference on Computer Aided Verification*, pages 333–339. Springer, 2016.
13. Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of computer and system sciences*, 57(1):94–124, 1998.
14. Hassan K. Khalil. *Nonlinear systems*. Prentice-Hall, Upper Saddle River, NJ, 2002.
15. Eric S. Kim, Murat Arcak, and Sanjit A. Seshia. A small gain theorem for parametric assume-guarantee contracts. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, HSCC '17, pages 207–216, New York, NY, USA, 2017. ACM.
16. Eric S. Kim, Sadra Sadraddini, Calin Belta, Murat Arcak, and Sanjit A. Seshia. Dynamic contracts for distributed temporal logic control of traffic networks. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pages 3640–3645. IEEE, 2017.
17. Scott C. Livingston, Pavithra Prabhakar, Alex B. Jose, and Richard M. Murray. Patching task-level robot controllers based on a local μ-calculus formula. In *2013*

IEEE International Conference on Robotics and Automation, pages 4588–4595, May 2013.
18. Rupak Majumdar, Nir Piterman, and Anne-Kathrin Schmuck. Environmentally-friendly gr (1) synthesis. *arXiv preprint arXiv:1902.05629*, 2019.
19. Shahar Maoz and Jan Oliver Ringert. Gr (1) synthesis for ltl specification patterns. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 96–106. ACM, 2015.
20. Pierluigi Nuzzo, Alberto L. Sangiovanni-Vincentelli, Davide Bresolin, Luca Geretti, and Tiziano Villa. A platform-based design methodology with contracts and related tools for the design of cyber-physical systems. *Proceedings of the IEEE*, 103(11):2104–2132, 2015.
21. Jan Outrata and Vilem Vychodil. Fast algorithm for computing fixpoints of galois connections induced by object-attribute relational data. *Information Sciences*, 185(1):114–127, 2012.
22. Francis J. Pelletier. The principle of semantic compositionality. *Topoi*, 13(1):11–24, 1994.
23. Tung Phan-Minh. Reactive contracts. `https://github.com/tungminhphan/reactive_contracts`, accessed April 30 2019, 2019.
24. Tung Phan-Minh. TLA+ specifications for an automated valet parking garage. `https://github.com/tungminhphan/reactive_contracts/blob/master/scenarios/AutomatedValetParking.tla`, accessed May 2 2019, 2019.
25. Nir Piterman and Amir Pnueli. Faster solutions of rabin and streett games. In *21st Annual IEEE Symposium on Logic in Computer Science (LICS'06)*, pages 275–284. IEEE, 2006.
26. Nir Piterman, Amir Pnueli, and Yaniv Saar. Synthesis of reactive (1) designs. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 364–380. Springer, 2006.

7 Appendix

Proposition 7. *If $\mathcal{C} = (\mathcal{A}, \mathcal{G}, \Delta, R)$ is a reactive contract, then the canonical form of $\mathcal{C}$ defined as $\mathcal{C}^\star := (\mathcal{A}, \mathcal{G}, \Delta, \Delta \Rightarrow R)$ satisfies $\mathcal{C}^\star = \mathcal{C}$.*

Proof. First, by Definition 7, it is clear that $\mathcal{E}_\mathcal{C} = \mathcal{E}_{\mathcal{C}^\star}$. Also, $M \in \mathcal{M}_\mathcal{C}$ is equivalent to

$$\begin{aligned} & \forall E \in \mathcal{E}_\mathcal{C}.\forall \sigma \in (M \cap E).\exists r \in R.\sigma \models^\rho r \wedge \Pi_\mathcal{A}(r) = \mathcal{U}_\mathcal{A}(\sigma) \\ \Leftrightarrow \quad & \forall E \in \mathcal{E}_{\mathcal{C}^\star}.\forall \sigma \in (M \cap E).\exists r \in R.\sigma \models^\rho r \wedge \Pi_\mathcal{A}(r) = \mathcal{U}_\mathcal{A}(\sigma) \\ \Leftrightarrow \quad & \forall E \in \mathcal{E}_{\mathcal{C}^\star}.\forall \sigma \in (M \cap E).\exists r \in \Delta \Rightarrow R.\sigma \models^\rho r \wedge \Pi_\mathcal{A}(r) = \mathcal{U}_\mathcal{A}(\sigma) \end{aligned}$$

which is equivalent to $M \in \mathcal{M}_{\mathcal{C}^\star}$. Note that the forward direction of the last "$\Leftrightarrow$" follows from the fact that $R \subseteq \Delta \Rightarrow R$. The reverse direction holds because for any $\sigma \in M \cap E$ where $E \in \mathcal{E}_{\mathcal{C}^\star} = \mathcal{E}_\mathcal{C}$, we have $\mathcal{U}_\mathcal{A}(\sigma) \in \Delta$, which implies that for any $r' \in \{r \mid r \in \mathfrak{R} \wedge \Pi_\mathcal{A}(r) \notin \Delta\}$, we obtain $\sigma \not\models^s \Pi_\mathcal{A}(r')$ by the initial disjointness of $\mathcal{A}$. By the first condition of Definition 8, we have $\sigma \not\models^\rho r'$. Therefore, the r that satisfies $\Delta \Rightarrow R$ must satisfy $r \in R$. □

Proposition 8. *Let there be canonical reactive contracts $\mathcal{C}_1$, $\mathcal{C}_2$ such that $\mathcal{C}_1 = (\mathcal{A}_1, \mathcal{G}_2, \Delta_1, R_1)$, $\mathcal{C}_2 = (\mathcal{A}_2, \mathcal{G}_2, \Delta_2, R_2)$ and $\mathcal{A}_1 \cup \mathcal{A}_2$ is initially disjoint, then the metatheoretic conjunction is $\mathcal{C}_1 \wedge \mathcal{C}_2 = (\mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{G}_1 \cap \mathcal{G}_2, \Delta_1 \cup \Delta_2, R_1 \cap R_2)$.*

Proof. Let $E_1 \in \mathcal{E}_{\mathcal{C}_1}$ and $E_2 \in \mathcal{E}_{\mathcal{C}_2}$,

$$\begin{aligned}
& \forall i \in \{1,2\}.E_i \in \mathcal{E}_{\mathcal{C}_i} \\
\Leftrightarrow \quad & \forall i \in \{1,2\}.\mathcal{U}_{\mathcal{A}_i}(E_i) \subseteq \Delta_i \\
\Leftrightarrow \quad & \forall i \in \{1,2\}.\mathcal{U}_{\mathcal{A}_1 \cup \mathcal{A}_2}(E_i) \subseteq \Delta_i \\
& (\mathcal{A}_1 \cup \mathcal{A}_2 \text{ is initially disjoint and conversely, } \Delta_i \text{ only consists of } \mathcal{A}_i\text{-signatures}) \\
\Rightarrow \quad & \forall i \in \{1,2\}.\mathcal{U}_{\mathcal{A}_1 \cup \mathcal{A}_2}(E_i) \subseteq \Delta_1 \cup \Delta_2 \\
\Leftrightarrow \quad & \forall i \in \{1,2\}.E_i \in \mathcal{E}_{\mathcal{C}_1 \wedge \mathcal{C}_2}
\end{aligned}$$

Hence

$$\mathcal{E}_{\mathcal{C}_1} \cup \mathcal{E}_{\mathcal{C}_2} \subseteq \mathcal{E}_{\mathcal{C}_1 \wedge \mathcal{C}_2} \tag{10}$$

Implementation-wise, if $M \in \mathcal{M}_{\mathcal{C}_1 \wedge \mathcal{C}_2}$ then for any environment E of $\mathcal{C}_1 \wedge \mathcal{C}_2$,

$$\begin{aligned}
& \forall \sigma \in (M \cap E).\exists r \in (R_1 \cap R_2).\sigma \models^\rho r \wedge \Pi_{\mathcal{A}_1 \cup \mathcal{A}_2}(r) = \mathcal{U}_{\mathcal{A}_1 \cup \mathcal{A}_2}(\sigma) \\
\Rightarrow \quad & \forall \sigma \in (M \cap E).\forall i \in \{1,2\}.\exists r \in R_i.\sigma \models^\rho r \wedge \Pi_{\mathcal{A}_1 \cup \mathcal{A}_2}(r) = \mathcal{U}_{\mathcal{A}_1 \cup \mathcal{A}_2}(\sigma) \\
\Leftrightarrow \quad & \forall \sigma \in (M \cap E).\forall i \in \{1,2\}.\exists r \in R_i.\sigma \models^\rho r \wedge \Pi_{\mathcal{A}_i}(r) = \mathcal{U}_{\mathcal{A}_1 \cup \mathcal{A}_2}(\sigma) \\
& \qquad (\text{since } r \in R_i, \text{ the assumption parts of } r \text{ must be from } \mathcal{A}_i) \\
\Leftrightarrow \quad & \forall \sigma \in (M \cap E).\forall i \in \{1,2\}.\exists r \in R_i.\sigma \models^\rho r \wedge \Pi_{\mathcal{A}_i}(r) = \mathcal{U}_{\mathcal{A}_i}(\sigma) \\
& \qquad (\text{since } \mathcal{A}_1 \cup \mathcal{A}_2 \text{ is initially disjoint}) \\
\Leftrightarrow \quad & \forall i \in \{1,2\}.\forall \sigma \in (M \cap E).\exists r \in R_i.\sigma \models^\rho r \wedge \Pi_{\mathcal{A}_i}(r) = \mathcal{U}_{\mathcal{A}_i}(\sigma) \\
& \qquad (\text{by interchanging universal quantifiers}) \\
\Rightarrow \quad & \forall i \in \{1,2\}.\forall E' : E' \in \mathcal{E}_{\mathcal{C}_i}.\forall \sigma \in (M \cap E').\exists r \in R_i.\sigma \models^\rho r \wedge \Pi_{\mathcal{A}_i}(r) = \mathcal{U}_{\mathcal{A}_i}(\sigma) \\
& \qquad (\text{by (10)}) \\
\Leftrightarrow \quad & \forall i \in \{1,2\}.M \in \mathcal{M}_{C_i}
\end{aligned}$$

Therefore $\mathcal{M}_{\mathcal{C}_1 \wedge \mathcal{C}_2} \subseteq \mathcal{M}_{\mathcal{C}_1} \cap \mathcal{M}_{\mathcal{C}_2}$. And thus we have $\mathcal{C}_1 \wedge \mathcal{C}_2 \preceq \mathcal{C}_1, \mathcal{C}_2$. To see that $\mathcal{C}_1 \wedge \mathcal{C}_2$ is really the meet, observe that that the contigency of any reactive contract $\mathcal{C}'$ that simultaneously refines $\mathcal{C}_1$ and $\mathcal{C}_2$ must at least contain $\Delta_1 \cup \Delta_2$ (otherwise there would be at least an environment of $\mathcal{C}_1$ or $\mathcal{C}_2$ that is not an environment of $\mathcal{C}'$, namely, the environment containing a behavior with the missing signature) and that the size of the environment set grows as we add more elements to the contigency. Dually, the reaction of $\mathcal{C}'$ must not contain anything that is not in $R_1 \cap R_2$. Because, it it does, then since $\mathcal{C}_1$ and $\mathcal{C}_2$ are canonical, the projection of a signature not in $R_1 \cap R_2$ must be in $\Delta_1 \cup \Delta_2$. But this implies that there is an implementation of $\mathcal{C}'$ having that signature but is not a shared implementation of both $\mathcal{C}_1$ and $\mathcal{C}_2$. □